Guaranteed interval integration for large initial boxes

Julien Damers[1], Luc Jaulin[1], and Simon Rohou[1]

[1]ENSTA-Bretagne, Lab-STICC UMR CNR 6285

Keywords: Interval analysis, Guaranteed integration, Lie groups, Symmetries

Introduction

A Lie group [4] is both an abstract and a smooth n-dimensional manifold so that multiplication and the inverse are both smooth. Lie groups have been introduced to model the continuous symmetries of differential equations. They are widely used for their resolution. The main idea of the new approach presented in this paper is to take advantage of the symmetries of the problem to extend one solution to get all other solutions.

Our main contribution is to show that the use of Lie symmetries can be combined with interval based methods to propagate uncertainties through differential equations [1]. More precisely, we will compute an enclosure of the solution to a differential equation assuming that the initial state is inside a box which may be large. The proposed method will be compared to existing methods such as CAPD [5] or DynIbex [2]. Some test-cases related to robotic applications illustrate the efficiency of our approach.

Problem

Most dynamical systems can be represented with a state equation such as $\dot{x} = f(x)$. When it is not possible to find an analytic solution for our equation we need an integration scheme to determine our solution. Conventional guaranteed interval integration libraries such as DynIbex or CAPD are able to compute a guaranteed solution without requiring much computing time when the initial condition is well described. However for some specific cases, for instance when attractors exist near the trajectory, these methods may have some difficulties. In many fields, for instance in robotics, the precision we have on our initial condition can be limited, thus the need of a tool for parameter estimation with larger initial boxes [3].

Our approach

Similarly to the variation of parameter method of Laplace, where one looks for a general solution knowing a particular solution of the differential equation, our method consists in first calculating precisely a trajectory using an integration tool such as CAPD or DynIbex applied with a given precise point as initial condition. We call this trajectory the *reference* (in red on Figure 1). Using the symmetries of the problem, we are able to calculate the solution for a different initial condition at a given time. We are able to do so without the need of applying conventional integration techniques i.e calculating each step, hence saving computation time.

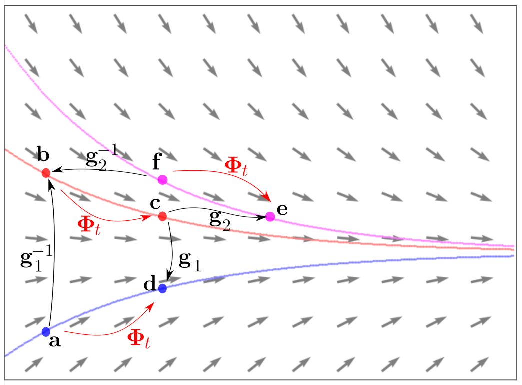

Figure 1: Integration principle illustration

Results

Consider the system following the equations below:

$$\begin{cases} \dot{x_1} = -x_1^3 - x_1 x_2^2 + x_1 - x_2 \\ \dot{x_2} = -x_2^3 - x_1^2 x_2 + x_1 + x_2 \end{cases}$$

and an initial condition $\mathbf{a_0} = \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix}$. We use a conventional guaranteed integration library to compute the trajectory associated with this system. Then we compare our approach to the result given by CAPD for an initial condition

$$[\mathbf{a_0}] = \begin{bmatrix} [0.4, 0.6] \\ [-0.1, 0.1] \end{bmatrix}$$

represented in red in Figure 2 and Figure 3.

As shown in Figure 3 we observe a bloating effect when using conventional interval integration method, in this case CAPD, the computation stops after a time $t = 0.76s$ because of the bloating effect. The approach presented in this paper is robust to the increase of the interval size as initial condition and is able to carry out the integration until the end set (here 7 seconds).

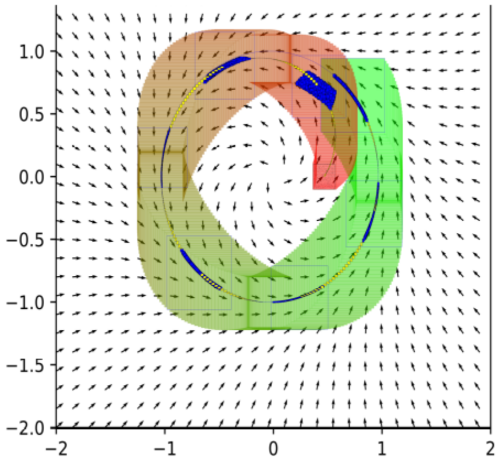

Figure 2: Integration using Lie symmetries

We also carried out the integration with 1000 particles picked randomly from the initial box $[\mathbf{a_0}]$ and calculated their positions at each second up to 7 seconds. The results obtained for both CAPD and our method is represented in blue on Figure 2. As we are only computing on points as initial condition it is possible to use CAPD for our particle cloud. We compared the computing time and as expected our approach achieves much better results on the computation time aspect compared to CAPD. It takes **297ms** with our method and **92811ms** using CAPD on an Intel Core I7-5700HQ @ 2.7GHz and 16GB of ram . This was expected as we don't need to calculate each step of the integration process but only the transformation for time step(s) enclosing our desired time slot.

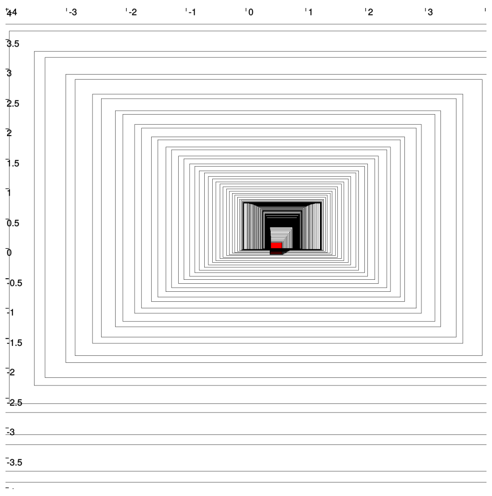

Figure 3: Integration using CAPD only

Acknowledgement

This work has been supported by the *French Government Defense procurement and technology agency* (DGA). It also benefited from the support of the project CONTREDO of the French National Research Agency (ANR).

References

[1] J. Alexandre dit Sandretto. Contraction, propagation and bisection on a validated simulation of ode. Lyon, France, 2016.

[2] J. A. dit Sandretto and A. Chapoutot. Validated explicit and implicit Runge–Kutta

methods. *Reliable Computing*, 22(1):79–103, Jul 2016.

[3] T. Raissi, N. Ramdani, and Y. Candau. Set membership state and parameter estimation for systems described by nonlinear differential equations. *Automatica*, 40:1771–1777, 2004.

[4] J. Serre. *Lie Algebras and Lie Groups*. Springer, 1965.

[5] D. Wilczak, P. Zgliczynski, P. Pilarczyk, M. Mrozek, T. Kapela, Z. Galias, J. Cyranka, and M. Capinsky. Computer assisted proofs in dynamics group, a c++ package for rigorous numerics, 2017.